**REMARKS**

Reconsideration of the application in view of the above amendments and following remarks is respectfully requested. In the above amendment, claims 1, 17-21, 27, 31, 32 and 33 are currently being amended, claim 30 is currently being canceled without prejudice, and claim 34 is currently being added. Therefore, claims 1-7, 9-29 and 31-34 are pending in the application.

Petition for Extension of Time

A Petition and Fee for a Two-Month Extension of Time to respond is submitted herewith to extend the period for response to October 30, 2006.

Objections to the Specification

Applicant has amended claim 27 to change the recited "sharing" algorithm to --shading-- algorithm. Applicant asserts that the specification provides proper support and antecedent basis for this subject matter.

Claim Rejections under 35 U.S.C. § 112

Claim 22 has been rejected under 35 U.S.C. 112, second paragraph, as allegedly being indefinite. Applicant respectfully traverses this rejection.

Specifically, the Examiner asserts that there is insufficient antecedent basis for the limitation "the program" in claim 22. However, claim 22 is dependent on claim 21, and claim 21 recites "a computer program" and then "the program comprising" in the preamble thereof. Therefore, the antecedent basis for the limitation "the program" in claim 22 is found in claim 21. As

such, the rejection should be withdrawn.

Claim Rejections under 35 U.S.C. § 102

Claims 1, 3, 5, 20, 25 and 26 have been rejected under 35 U.S.C. 102(b) as being anticipated by U.S. Patent No. 6,016,150 to Lengyel et al. ("Lengyel et al."). Furthermore, claims 19 and 33 have been rejected under 35 U.S.C. 102(b) as being anticipated by U.S. Patent No. 5,867,166 to Myhrvold et al. ("Myhrvold et al."). Applicant respectfully traverses these rejections.

The system disclosed in Myhrvold et al. relies on so-called "multi-layered image rendering" architecture as described below:

> "In our system, multiple independent image layers may be composited together at video rates to create the output video signal. These image layers, which we refer to as generalized sprites, or gsprites, can be rendered into and manipulated independently. The system will generally use an independent gsprite for each non-interpenetrating object in the scene. This allows each object to be updated independently, so that object update rate can be optimized based on scene priorities. For example, an object that is moving in the distant background may not need to be updated as often, or with as much accuracy, as a foreground object." (Myhrvold et al., column 7, lines 46-56).

In the system of Myhrvold et al., the scene is broken into multiple image layers or gsprites. Each gsprite contains a *non-interpenetrating* object, that is, an object that does not penetrate another object, allowing each *non-interpenetrating* object to be updated independently. Non-interpenetrating objects in the scene can be assigned to separate and independent gsprites, if the system has sufficient memory and computing capacity. On the other hand, *interpenetrating* or *self-occluding* objects cannot be assigned to separate gsprites, but they will be assigned to a single gsprite and then processed as a single

gsprite, as described below:

> "The image preprocessor determines the configuration of gsprites for the image (244). This step involves finding how to map potentially visible objects to gsprites. As part of this process, the image preprocessor 104 allocates gsprites, which includes creating a gsprite data structure to store image data corresponding to one or more potentially visible objects. If processing resources allow, each non-interpenetrating object in the scene is assigned to an independent gsprite. Interpenetrating or self-occluding objects may be processed as a single gsprite." (Myhrvold et al., column 14, lines 3-12).

Since the system relies on the "multi-layered image rendering" architecture and accesses a list of gsprite to be displayed for the current frame, the image preprocessor determines the depth order of gsprites when determining the gsprite configuration. The image preprocessor sorts gsprites in depth order and stores *this* depth data in the gsprite data structure, as described below:

> "During the display process, the image processor accesses a list of gsprites to be displayed for the current frame. In the process of determining the gsprite configuration, the image preprocessor determines the depth order of gsprites (280). As noted above, one object is preferably assigned to a gsprite. However, the image preprocessor can assign more than one object to a gsprite, for example, to accommodate processing constraints of a particular image processor being used in the system. The image preprocessor sorts objects in Z-order, i.e. in distance from the viewpoint. In addition to sorting objects, it sorts gsprites in depth order as well and stores this depth data in the gsprite data structures. (Myhrvold et al., column 15, lines 23-34).

In other words, in the system of Myhrvold et al., each

gsprite has its own depth value so that the gsprites can be sorted in depth order. This assumption comes from the multi-layered image rendering architecture which the system inherently relies on. This is the reason why the system of Myhrvold et al. imposes such a restriction that *interpenetrating* or *self-occluding* objects cannot be assigned to separate independent gsprites but should be assigned to a *single* gsprite.

The system disclosed in Lengyel et al. also relies on the *sprite* architecture as described below:

> "The invention comprises a method for performing lighting and shading operations in a real time graphics system by factoring a shading model into image layers and combining the image layers with an image compositor. The invention also includes a sprite compositor for combining factored layers into an output image. In the method, a layered graphics rendering pipeline factors terms in a shading model into separate layers (sprites), renders the layers independently, and uses a sprite compositor to combine the terms of the shading model." (Lengyel et al., column 3, lines 56-65).

On the other hand, according to Applicant's amended independent Claim 1, a grouping unit selects rendering strategy according to characteristics of input three-dimensional objects and groups the three-dimensional objects into groups in such a manner that the three-dimensional objects to which the same rendering strategy is applied are grouped into the same group. A rendering processing unit derives a three-dimensional subspace which contains the three-dimensional objects belonging to the same group to be an independent rendering unit and performs rendering processing individually on the subspace by applying the group by group different rendering strategy, and generates independent image data for each subspace. The grouping unit

groups the three-dimensional objects into groups in such a manner that the three-dimensional subspaces, each of which contains at least one three-dimensional object belonging to the same group, are allowed to spatially overlap one another.

Since the apparatus of Claim 1 does not adopt the multi-layered image rendering, the apparatus does not impose such a requirement on grouping that the subspaces derived from the groups should be sorted in the depth order. Therefore, one subspace that contains the objects belonging to one group can overlap another subspace that contains the objects belonging to another group.

Examples of this technical feature of claim 1 are disclosed in Applicant's Specification in Figs. 3A-3B, 4A-4B, 12A-12B, 13A-13B, 17A-17B and 18A-18B, and the corresponding discussion thereof. For a better understanding of this feature, the example shown in Figs. 17A-17B and 18A-18B of Applicant's Specification will now be explained.

Referring to Figs. 17A-17B, B-boxes 200 and 202 each contain a high-speed object. The B-boxes 200 and 202 are sorted in the same group according to their motion characteristics. Then a brick 204 is derived to be a subspace which contains the two B-boxes 200 and 202 belonging to the same group. Consider a situation in which after the brick 204 is determined, another B-box 206 containing a low-speed object comes into the brick 204. Because of the difference in motion characteristics, the B-box 206 is sorted in a group different from the group that the two B-boxes 200 and 202 belong to. Therefore, as shown in Figs. 18A-18B, a new brick 208 is derived to be a subspace which contains the B-box 206 with its low-speed object. The brick 208 is overlapped by the brick 204 which contains the two B-boxes 200 and 202 which both contain a high-speed object. For a detailed

description, see Applicant's Specification, page 45, line 21 to page 47, line 13.

When the objects are grouped in such a manner that the objects to which the same rendering strategy is applied are grouped into the same group, the situation would happen in which a subspace derived from one group has an overlap with another subspace derived from another group. The embodiments of the present invention allow the situation in which there is overlapping between the subspaces derived from the groups. Since the subspaces derived from the groups are allowed to spatially overlap one another, it gives flexibility in grouping the objects according to the rendering strategy.

In Item 8 of Office Action, the Examiner applies "layers (sprites)" disclosed in Lengyel et al. to the subspaces in Applicant's previous Claim 1. However, "layers" cannot spatially overlap one another but "layers" can be sorted in the depth order, from the front to the back. On the other hand, the subspaces in Applicant's Claim 1 can spatially overlap one another and the subspaces are not required to be sorted in the depth order. Therefore, "layers (sprites)" disclosed in Lengyel are totally different from the subspaces in Applicant's Claim 1. The same is applied to "gsprites" disclosed in Myhrvold et al. which are also layers and cannot spatially overlap one another.

In Item 15 of Office Action, the Examiner applies "chunks" disclosed in Myhrvold et al. to the subspaces in Applicant's previous Claim 19. However, "chunks" are pixel regions (32x32 pixels in one specific implementation) and are not three-dimensional spaces. On the other hand, the subspaces in Applicant's amended Claim 1 are three-dimensional spaces and therefore the subspaces can spatially overlap one another. The Examiner also cited the sentence that "Geometry that overlaps a

chunk boundary is preferably referenced in each chunk it is visible in." in Myhrvold et al. and applies the expression "Geometry that overlaps a chunk boundary" to "the subspaces which overlap one another" in Applicant's previous Claim 19. However, the sentence "Geometry that overlaps a chunk boundary is preferably referenced in each chunk it is visible in" means that geometry which extends across a chunk (specifically a 32x32 pixel region) boundary is referenced in each chunk where the geometry exists. The cited sentence has nothing to do with the context that the three-dimensional subspaces spatially overlap one another, as claimed in Applicant's amended Claim 1.

The systems disclosed in Myhrvold et al. and Lengyel et al., which both rely on sprite or multi-layered image rendering architecture, do not disclose nor suggest the above-mentioned technical feature of Applicant's amended Claim 1. Therefore, the rejections of amended Claim 1 should be withdrawn. Furthermore, the rejections of amended independent Claims 19 and 20 should also be withdrawn for similar reasons, and the rejections of dependent Claims 3, 5, 25 and 26 should also be withdrawn for at least these same reasons due to their dependency on claim 1.

In addition, Applicant asserts that the above arguments also apply to amended independent Claims 17, 18, and 21. As such, the rejection of those claims should also be withdrawn for similar reasons.

Turning now to amended independent claim 33, the Examiner pointed out that Lengyel et al. does not expressly disclose per-pixel Z values (Item 47 of the Office Action), but that Myhrvold et al. discloses the image data having per-pixel Z values (Item 16e of the Office Action). The Examiner also has an opinion that Myhrvold et al. discloses "Z-merge processing of the image data generated for each subspace according to the per-pixel Z values".

(Item 16f of the Office Action). Applicant believes, however, that this analysis indicates a misunderstanding of the relationship between the operation performed by the tiler 200 and the operation performed by the gsprite engine 204, disclosed in Myhrvold et al.

Specifically, in the tiler 200, the rendering process is performed by a unit of chunk and the geometry visible in each chunk is rendered by using Z-buffer algorithm according to pixel Z-values. The chunk may be touched by a plurality of objects as shown in FIG. 18B of Myhrvold et al. Each non-interpenetrating object is assigned to a separate and independent gsprite and interpenetrating or self-occluding objects are assigned to a single gsprite. If the chunk is touched by a plurality of gsprites, each gsprite is separately and independently rendered in the chunk. In other words, the Z-buffer algorithm is applied to each gsprite individually and each non-interpenetrating object assigned to each gsprite is rendered by performing hidden surface removal based on the pixel Z-values. If interpenetrating or self-occluding objects are assigned to a gsprite, the interpenetrating or self-occluding objects assigned to the gsprite are rendered by performing hidden surface removal based on the pixel Z-values.

However, Z-merge processing is not performed in the chunk between the individual gsprites. If Z-merge processing is performed between the individual gsprites by the tiler 200 and a final image in which the gsprites are composed is generated, the gsprite engine 204 cannot operate the rendered gsprites separately. The purpose to introduce gsprites is to render gsprites independently, which enables each non-interpenetrating object or interpenetrating or self-occluding objects represented by each gsprite to be rendered at different resolution and

updated at varying rates (See Myhrvold et al., column 49, lines 18-20). The tiler 200 renders each gsprite independently by a unit of chunk and generates the rendered image data for each gsprite. The gsprite engine 204 use the rendered image data for each gsprite and performs an affine transform for the gsprite. Then the gsprite engine 204 composites the affine-transformed gsprites to generate a final image to be displayed (See Myhrvold et al., column 61, lines 5-20). The composition of the affine-transformed gsprites is easily performed in the compositing buffers 210, since the gsprites are sorted beforehand in depth order.

On the other hand, according to Applicant's amended independent Claim 33, a grouping unit groups input three-dimensional objects into groups. A rendering processing unit which derives a subspace which contains at least one three-dimensional object belonging to the same group to be an independent rendering unit and performs rendering processing individually on the subspace, and generates independent image data having per-pixel Z values indicating depth information on a pixel by pixel basis for each subspace. A consolidation unit generates final output image data to be displayed by consolidating the image data having per-pixel Z values generated for each subspace.

According to Applicant's amended Claim 33, independent image data having per-pixel Z values is generated for each subspace and the image data having per-pixel Z values generated for each subspace is consolidated so as to generate a final image data to be displayed. In Myhrvold et al., the rendered image for each gsprite does not have per-pixel Z values but the rendered gsprites are only composed in the depth order so as to generate a final image to be displayed. The gsprite engine 204 in Myhrvold

et al. does not need per-pixel Z values for each gsprite, since the gsprites can be sorted in the depth order. In the apparatus of Applicant's amended Claim 33, the consolidation unit needs per-pixel values of the image data generated for each subspace, since the subspaces may spatially overlap one another and therefore Z-merge processing may be needed.

The systems disclosed in Myhrvold et al. and Lengyel et al., which both rely on sprite or multi-layered image rendering architecture, do not disclose nor suggest the above-mentioned technical feature of Applicant's amended Claim 33. Therefore, the rejections of amended independent Claim 33 should be withdrawn.

In addition, Applicant asserts that the above arguments also apply to dependent Claims 32 and 34. As such, the rejection of claim 32 should also be withdrawn for similar reasons, and new claim 34 should be allowed.

Claim Rejections under 35 U.S.C. § 103

Claims 2, 4, 6, 21 and 22 have been rejected under 35 U.S.C. 103(a) as being unpatentable over Lengyel et al. Applicant respectfully traverses these rejections.

As mentioned above, Applicant asserts that the above arguments for amended Claim 1 also apply to amended independent Claim 21. As such, the rejection of Claim 21 should also be withdrawn for similar reasons. Furthermore, the rejections of dependent Claims 2, 4, 6, and 22 should also be withdrawn for at least these same reasons due to their dependency on their respective independent claims.

Claims 28 and 29 have been rejected under 35 U.S.C. 103(a) as being unpatentable over Lengyel et al. in view of U.S. Patent No. 5,986,659 to Gallery et al. ("Gallery et al."). Applicant

respectfully traverses these rejections.

The rejections of dependent Claims 28 and 29 should be withdrawn for at least the above reasons due to their dependency on their respective independent claims.

Claims 9-14, 17, 18, 23, 24 and 27 have been rejected under 35 U.S.C. 103(a) as being unpatentable over Lengyel et al. in view of Applicant's allegedly admitted prior art Kwan Liu Ma, James S. Painter, Charles D. Hansen, Michael F. Krogh, "A data distributed, parallel algorithm for ray-traced volume rendering," October 25, 1993, Proceedings of the 1993 Symposium on Parallel Rendering, p. 15-22. ("Ma et al."). Applicant respectfully traverses these rejections.

As mentioned above, Applicant asserts that the above arguments for amended Claim 1 also apply to amended independent Claims 17 and 18. As such, the rejection of Claims 17 and 18 should also be withdrawn for similar reasons. Furthermore, the rejections of dependent Claims 9-14, 23, 24 and 27 should also be withdrawn for at least these same reasons due to their dependency on their respective independent claims.

Claims 15 and 16 have been rejected under 35 U.S.C. 103(a) as being unpatentable over Lengyel et al. in view of Ma et al. and in further view of Jerrell Watts, Stephen Taylor, "A Practical Approach to Dynamic Load Balancing," March 1998, IEEE Transactions on Parallel and Distributed Systems, v. 9 n. 3, p. 235-248. Applicant respectfully traverses these rejections.

The rejections of dependent Claims 15 and 16 should be withdrawn for at least the above reasons due to their dependency on their respective independent claims.

Claims 7, 30, 31 and 32 have been rejected under 35 U.S.C. 103(a) as being unpatentable over Lengyel et al. in view of

Myhrvold et al. Applicant respectfully traverses these rejections.

Claim 30 has been canceled without prejudice in the above amendment, and so that rejection is now moot.

As mentioned above, Applicant asserts that the above arguments also apply to dependent Claim 32. As such, the rejection of claim 32 should also be withdrawn for similar reasons. Furthermore, the rejections of dependent Claims 7 and 31 should be withdrawn for at least the above reasons due to their dependency on their respective independent claims.

Fees Believed to be Due

Fees have previously been paid in this application for a total of 32 claims with 7 claims being independent claims. The above amendment has resulted in there now being a total of 32 claims with 7 claims being independent claims. Thus, no extra claims fees are believed to be due.

**CONCLUSION**

In view of the above, Applicant submits that the pending claims are in condition for allowance. Should there remain any outstanding issues that require adverse action, it is respectfully requested that the Examiner telephone Richard E. Wawrzyniak at (858)552-1311 so that such issues may be resolved as expeditiously as possible.

Respectfully submitted,

Date: 10/16/06

Richard E. Wawrzyniak
Attorney for Applicant(s)
Reg. No. 36,048
(858) 552-1311

**Address all correspondence to:**
Richard E. Wawrzyniak, Esq.
FITCH, EVEN, TABIN & FLANNERY
120 So. LaSalle Street, Suite 1600
Chicago, Illinois 60603
Customer No. 22242
Telephone No.: (858) 552-1311
Facsimile No.: (858) 552-0095